Genetic Algorithm Optimization of Inlet Geometry for a Hypersonic Jet Engine with Mode Transition

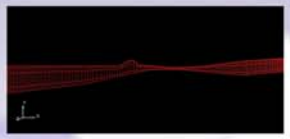
Ashley Micks
NASA Academy Research Associate, Aeronautical and Astronautical Engineering, MIT

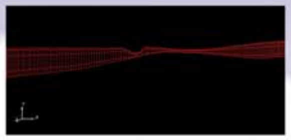
Dr. Meng-Sing Liou
Staff Mentor, NASA-GRC Aerospace Engineer, Research and Technology

How do you design a genetic algorithm to optimize geometry?

What are the alleles?

- **coordinates** *of key points along the inlet walls*
- **"bias"** *numbers used to determine the shape of the wall between the key points*
- **"tightness"** *numbers also used to determine wall shape between the key points*


inlet profile at bias = -20 for the bumped-out section, bias = 0 elsewhere


bias = 20 for the same section, bias = 0 elsewhere

tightness = -10 for the same section, tightness = 0 elsewhere

tightness = 10 for the same section, tightness = 0 elsewhere

tightness = 20 for the same section, tightness = 0 elsewhere

How does crossover work?

- **Setup:** *The chromosomes are lists of the x and y coordinates of key points along the walls, followed by tightness values for each wall section where Hermite interpolation is used (as opposed to linear), and then bias values for each of those sections. There is a set probability that any two parents will experience crossover.*
- **Problem:** *That means the standard technique--choosing a point along the chromosome and swapping everything after that between the two parents--won't mix thing up like it would if the alleles were arranged more randomly along the chromosome.*
- **Solution:** *When crossover occurs, go through each allele and decide randomly whether or not it will be swapped between parents. The individuals that result from this series of swaps are the two children.*

How does mutation work?

- **Setup:** *Mutation happens to the children after they are produced by crossover. There is a set probability that a given child will experience mutation.*
- **Problem:** *The alleles aren't binary, so a mutation can't mean simply changing a random digit in the chromosome from a 0 to a 1 or vice versa.*
- **Solution:** *When mutation occurs, choose a random allele on that individual and change its value by a random amount of up to 5 inches if it's a coordinate, or up to 0.5 if it's a bias or tightness value.*

Other characteristics of the algorithm

- **Elitism:** *The best individual (the one with the highest total pressure recovery) in the latest generation* automatically goes into the next generation, unaltered by crossover or mutation.
- **Constraints:** *An inlet's lower wall must not intersect its upper wall, and the lip of its cowl must come far enough forward that it catches the shock produced by the lip of the ramp.*
- **Redos:** *If a new individual violates a design constraint, the process that produced the individual (crossover, mutation, or random generation in the case of the first generation) is redone until a functioning individual is produced.*

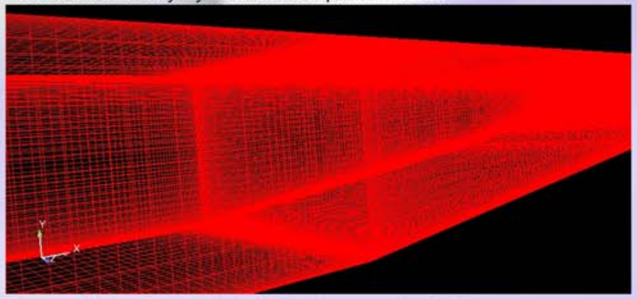
What else is needed to run the algorithm?

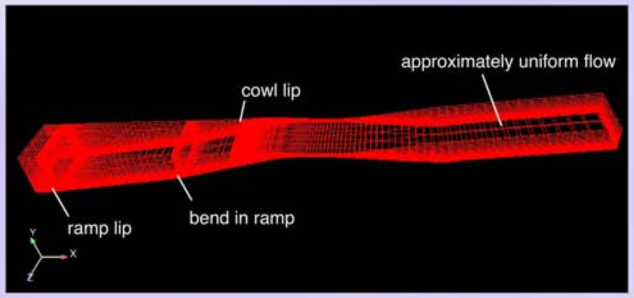
Overflow

- **Overflow requires specific input files** *for it to do the calculations needed to find an inlet's total pressure recovery.*
- **Grid:** *Overflow needs a 3D mesh grid representing the inlet's geometry, written in Plot3D format (a text file with the grid's dimensions followed by the x, y, and z coordinates of all the points where the gridlines intersect).*
- **Boundary conditions:** *Another input file needs to tell Overflow what the conditions at the borders of the grid are like--whether there's friction, whether there can be flow though the border, etc.*
- **Overflow doesn't output the total pressure recovery.** *It outputs density, velocity, and a list of other values that the algorithm must use to calculate total pressure recovery.*

Grid generation

- **A program separate from the algorithm generates a Plot3D grid for Overflow, using the information from a given inlet's chromosome.**
- **Interpolation:** *Since the chromosome only has coordinates for certain points on the inlet walls, the grid generator must interpolate between them in order to find piecewise functions defining the walls. Some wall sections are linear, but others are curved. The curved sections use Hermite interpolation, which finds a cubic function that fits the given points while accounting for the bias and tightness for that section.*
- **Gridline concentration:** *The grid needs to be finer where the flow is more complicated. This includes at and after the lips of the cowl and ramp, and along the walls. The lips are where shock waves begin, and the walls are where the boundary layers and flow separation occur.*


This is part of the baseline inlet design's grid. Note the concentration of gridlines near the walls and the ramp lip, where the lower wall changes slope. A grid at least this dense is necessary where the flow cannot be approximated as uniform.



A full-length view of the baseline design, showing appropriate grid concentration along the inlet.

Why is geometry important?

- **Wall angles affect shock waves,** *and a stronger shock has a greater increase in entropy across it, which means a greater loss of total pressure and ability for the flow to do work.*
- **Geometry provides passive control of boundary layer thickness and shock strength,** *which are key factors in total pressure loss.*
- **Flow separation** *due to poorly shaped walls can be catastrophic for total pressure recovery.*

Work done so far

- Researched genetic algorithms and the hypersonic inlet to be optimized
- Researched how to code in Fortran (no previous experience)
- Installed Force and EnSight, after discovering a bug in one version of EnSight
- Set up an account on the Columbia supercomputer
- Installed Overflow on Columbia
- Ran test cases in Overflow and EnSight to gain familiarity with the programs
- Wrote a grid generation program
- Refined the grid generation program through plotting on EnSight
- Planned methods for crossover, mutation, and other processes the algorithm would perform
- Wrote the genetic algorithm, except for the call to Overflow to find the total pressure recovery

Work yet to do

- Learn how to call Overflow from a Fortran program
- Gain experience running small batch jobs to Columbia, to prepare for the large batch jobs needed to find total pressure recovery for each of about 100 individuals per generation for about 40 generations
- Upload all necessary programs and input files to Columbia
- Run the algorithm
- Analyze the results, looking at the individuals produced and the "optimal" individual with the best total pressure recovery in the last generation
- Refine the algorithm, grid generator, and input files as necessary

Acknowledgments

Special thanks to Meng-Sing Liou, May-Fun Liou, Dave Saunders, MaryJo Long-Davis, Jay Horowitz, and Mary Vickerman for their mentorship and support. Thanks also to the Massachusetts Space Grant Consortium.

Massachusetts Institute of Technology